REMARKS

Claims 1-38 are currently pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejection:

The Office Action rejected claims 1-38 under 35 U.S.C. § 103(a) as being unpatentable over Cheesman et al. (U.S. Patent 6,680,933) (hereinafter "Cheesman") in view of Maresco (U.S. Patent 6,418,458). Applicants respectfully traverse this rejection in light of the following remarks.

Regarding claim 1, contrary to the Examiner's assertion, Cheesman in view of Maresco fails to teach a primary scheduler which is executable to schedule *requests* for networked data resources.

Cheesman teaches a telecommunications switch for switching protocol data units between communications links connecting the telecommunications switch into a communications network. The telecommunications switch of Cheesman is operable to switch protocol data units of a plurality of services and includes a structure of queues and schedulers associated with one of the communications links. The queues and schedulers of Cheesman's switch are arranged to provide, for each service, one of class-based traffic management, flow-based traffic management, and traffic management that is both class-based and flow-based. *See* Cheesman, column 1, lines 45-60 and column 10, lines 51-67.

Maresco teaches a task queue and a work crew class that contains workers to complete the tasks. Maresco's system creates threads to connect the workers to the tasks in the task queue, and manages the creation of threads to prioritize the execution of the tasks using object oriented programming techniques (Maresco, column 1, lines 42-57; column 2, lines 25-38).

Thus, Cheesman pertains to the low-level switching of data packets according to communication protocols, such as for ATM cells or IP packets (Cheesman -- col. 5, lines 35-48; col. 8, lines 8-13), and Maresco pertains to the creation of threads to prioritize the execution of tasks in an object oriented system (Maresco, column 1, lines 42-57; column 2, lines 25-38). Neither Cheesman nor Maresco has anything to do with scheduling requests for networked data resources. Cheesman does not teach that the protocol data units handled by his switch are requests for networked data resources. And the threads in Maresco are clearly not requests for networked data resources. Thus, Cheesman and Maresco clearly do not teach or suggest anything about scheduling requests for networked data resources.

In response to the above argument, the Examiner argues, in the Response to Arguments section of the Final Office Action, that Cheesman teaches, "a request of different task and queue within the network, citing column 2, lines 34-40 of Cheesman. However, the cited portion describes how Cheesman's multiple queues and schedulers handle different types of services. For example, Cheesman describes how his system allows a Carrier to support a service that requires "a combination of class-based queuing and flow-based queuing" (Cheesman, column 2, lines 23-34). The Examiner's cited passage does not mention anything about scheduling *requests* for networked data resources. Instead, Cheesman is describing how his switch is capable of scheduling protocol data units based on a combination of different class and flow requirements. As noted above, nowhere does Cheesman mention *requests for networked data resources*.

Further regarding claim 1, Cheesman in view of Maresco does not teach a secondary scheduler that is executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner and send the requests to the primary scheduler in a thread-safe manner. The Examiner appears to be relying on Cheesman's priority scheduler (P) and weighted fair queuing (WFQ) scheduler (W) of scheduler 144(a) in Fig. 8 and as described at column 11, lines 5-30 of Cheesman. However, Cheesman's scheduler 144(a) receives protocol data units from queues 142, not requests for networked data resources from a multi-threaded application. As discussed

above, Cheesman's teachings pertain to a telecommunication switch that switches protocol data units. Thus, the scheduler in Cheesman is scheduling the switching of data units according to a communication protocol. The data in Cheesman are not scheduled as requests for networked data resources. Scheduling for low-level switching of protocol data units has nothing to do with scheduling requests for networked data resources.

Moreover, the protocol data units received by the scheduler in Cheesman are received from queues 142, not from a multi-threaded application. The multiple queues in Cheesman are not, and in no way imply, any type of multi-threaded application. In fact, Cheesman's teachings all apply to low-level communication protocol levels in a switch fabric, not to receiving a plurality of requests from a multi-threaded application.

In response to the above argument, the Examiner asserts "Cheesman teaches the use of a multi threaded application" and cites both FIG. 7 and column 11, lines 5-13 of Cheesman. However, FIG. 7 has nothing whatsoever to do with a multi-threaded application. Instead, FIG. 7 illustrates a flow diagram of Cheesman's enqueue task 126. Enqueue task 126 is a part of processor 124 that is a part of egress processor 114 (Cheesman, FIG. 6 and 7; column 9, lines 5-6, column 10, lines 1-5). Enqueue task 126 is part of Cheesman's system for scheduling protocol data units and clearly does not represent a multi-threaded application. Cheesman does not describe enqueue task 126 has having multiple threads. Nothing in FIG. 7 represents a multi-threaded application. The Examiner seems to be referring to the multiple logical data paths illustrated as part of the flow diagram of FIG. 7. However, multiple data paths do not define or imply a multi-threaded application. Instead, the multiple data paths illustrated in FIG. 7 represent how enqueue task 126 determines the type of service and flow for a protocol data unit (Cheesman, column 10, lines 11-32).

The Examiner admits that Cheesman does not teach a thread safe system and therefore that Cheesman does not teach a secondary scheduler receiving requests in a thread-safe manner and sending the requests to a primary scheduler in a thread-safe manner. The Examiner relies upon Maresco to teach a "thread safe method for working

with objects in a multi-threaded environment." However, Maresco does not teach or suggest anything about receiving or sending requests for networked data resources in a thread-safe manner. Instead, Maresco is concerned with safely *creating* threads for task execution is a computer system (Maresco, column 1, lines 42-57; column 2, lines 25-38).

Maresco teaches nothing about a scheduler receiving and sending requests in a thread-safe manner. Just because Maresco mentions thread creation for task execution does not imply that Maresco suggests receiving and sending requests by schedulers in a thread-safe manner. As discussed above, neither Cheesman nor Maresco has anything to do with scheduling requests from a multi-threaded application, let alone receiving and sending requests for scheduling in a thread-safe manner. Furthermore, the protocol data units scheduled in Cheesman's switch are simply low-level data units and do not have any sort of multi-threaded nature for which thread safety would be a concern. It would make no sense to apply any sort of thread safety techniques to Cheesman's telecommunications switch. Thus, Cheesman in view of Maresco clearly does not teach a secondary scheduler that is executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner and send the requests to the primary scheduler in a thread-safe manner.

In the Response to Arguments section of the Final Office Action, the Examiner states that Applicants previously argued that Maresco fails to teach a thread safe system and cites column 2, lines 30-32 of Maresco. The Examiner has apparently misunderstood Applicants' previous remarks. As described above, Applicants argument is that Cheesman in view of Maresco fails to teach receiving a plurality of requests from a multi-threaded application in a thread-safe manner. Just because Maresco teaches a manageable, efficient, thread-safe method for working with objects in a multi-threaded environment does not imply that Maresco teaches or suggests anything regarding a secondary scheduler executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner. The Examiner has failed to provide any evidence or reasoning to rebut this argument.

Additionally, the Examiner has failed to state a proper *prima facie* rejection under § 103(a). Specifically, the Examiner does not show how the prior art suggests or motivates the Examiner's suggested modification or combination of the teachings of Cheesman and Maresco. This is a basic requirement of all obviousness rejections (*see*, M.P.E.P. § 2143). Furthermore, "[w]hen the motivation to combine the teachings of the references is not immediately apparent, it is the duty of the examiner to explain why the combination of the teachings is proper" (*see*, M.P.E.P. § 2142, paragraph 5). The Examiner has not provided any motivation to combine the teachings of Cheesman and Maresco. Since the Examiner has not satisfied this basic requirement of a *prima facie* rejection under § 103(a), the rejection is clearly improper. **Even though this argument was presented in the response to the previous Office Action, the Examiner has still failed to provide any rebuttal argument or any motivation for the proposed combination of Cheesman and Maresco.**

Moreover, as discussed above, it would make no sense to apply any sort of thread safety techniques to Cheesman's telecommunications switch since the protocol data units scheduled in Cheesman's switch are simply low-level data units and do not have any sort of multi-threaded nature for which thread safety would be a concern. Maresco specifically states that his system "uses object-oriented programming to provide a manageable, efficient, thread-safe method for working with objects in a multi-threaded environment" (Maresco, column 2, lines 30-32). Cheesman does not teach or suggest that his system is, or would gain any advantage from being, multi-threaded. Thus, since Cheesman's system does not have anything to do with, nor have any need for, thread safety, there is no reason or motivation to apply any multi-threading teachings from Maresco to Cheesman's system.

Furthermore, even if the combination was made, the Examiner's suggested combination of Cheesman and Maresco would not result in a system that included a primary scheduler which is executable to schedule requests for networked data resources; and a secondary scheduler that is executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner and send the requests to the primary

scheduler in a thread-safe manner. Instead, the proposed combination of Cheesman and Maresco would result only a protocol data unit scheduling switch as taught by Cheesman that included creating multiple threads, as taught by Maresco, to handle the various queuing and scheduling tasks of Cheesman.

For at least the above reasons, the rejection of claim 1 is clearly unsupported by the cited art and should be withdrawn. Similar arguments also apply for independent claims 15 and 27.

In further regard to claim 15, Cheesman in view of Maresco, contrary to the Examiner's assertion, fails to teach or suggest executing the management requests on the managed objects after scheduling the management requests in the primary queue. As discussed above regarding claim 1, Cheesman teaches a telecommunications switch for switching protocol data units between communications links connecting the telecommunications switch into a communications network. The telecommunications switch of Cheesman is operable to switch protocol data units of a plurality of services and includes a structure of queues and schedulers associated with one of the communications links. The queues and schedulers of Cheesman's switch are arranged to provide for each service one of class-based traffic management, flow-based traffic management, and traffic management that is both class-based and flow-based. *See* Cheesman, column 1, lines 45-60; column 10, lines 51-67. Maresco teaches a task queue and a work crew class that contains workers to complete the tasks. Maresco's system creates threads to connect the workers to the tasks in the task queue, and manages the creation of threads to prioritize the execution of the tasks using object oriented programming techniques (Maresco, column 1, lines 42-57; column 2, lines 25-38).

Thus, Cheesman pertains to the low-level switching of data packets according to communication protocols, such as for ATM cells or IP packets (Cheesman -- col. 5, lines 35-48; col. 8, lines 8-13), and Maresco pertains to the creation of threads to prioritize the execution of tasks in an object oriented system (Maresco, column 1, lines 42-57; column 2, lines 25-38). Neither Cheesman nor Maresco has anything to do with executing

management requests on managed objects. Cheesman does not teach that the protocol data units handled by his switch are management requests for managed objects. And the threads in Maresco are clearly not management requests for managed objects. Thus, Cheesman and Maresco clearly do not teach or suggest anything about executing management requests on managed objects.

The Examiner cites column 11, lines 25-30 of Cheesman. However, this portion of Cheesman only describes how Cheesman's priority and WFQ schedulers allocate bandwidth to queues. Allocating bandwidth to protocol data units does not teach executing management requests on managed objects. In the Final Action, the Examiner fails to provide any argument or explanation regarding his interpretation of Cheesman. Cheesman is not concerned with executing management requests on managed object, but instead, as noted above, is concerned with switching protocol data unites between communication links using a system of queues and schedulers. Cheesman makes no mention, whether at the Examiner's cited portion or elsewhere, of executing management requests on managed objects.

Additionally, the arguments presented above regarding the rejection of claim 1 also apply to claim 15. Thus, for at least the reasons given above, the rejection of claim 15 is clearly not supported by the prior art and its removal is respectfully requested. Similar arguments as presented above regarding claim 15 also apply to claim 27.

Applicants also specifically traverse the Examiner's rejection of each of the dependent claims. In regard to each dependent claim, Applicants see little, if any, relevance of the sections of Cheesman and Maresco cited by the Examiner.

Regarding claim 2, The Examiner contends that Maresco teaches a system wherein the primary scheduler is single-threaded and cites column 2, lines 47-52 of Maresco. However, the cited passage makes no mention of a primary scheduler being single-threaded. Instead, the cited passage only describes how a computer system, suitable for implementing Maresco's invention, may operate under the control of an

operating system that may execute one or more threads. Thus, Maresco is teaching that his invention may be implemented on a multi-threaded OS. However, a general statement regarding the multi-threaded capability of an OS does not imply or suggest anything regarding the single or multi-threaded nature of any particular component of Maresco's or Cheesman's inventions. Thus, the rejection of claim 2 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 3, contrary to the Examiner assertion, Cheesman in view of Maresco fails to teach or suggest a system including a secondary scheduler that is multi-threaded. The Examiner cites column 1, lines 32-38 and column 2, lines 47-52 of Maresco. The cited passages do not teach or suggest anything about a secondary scheduler; nor about a secondary scheduler being multi-threaded. Instead, the first cited passage (column 1, lines 32-38) describes a need for a managing efficiency and thread safety in a multi-threaded programming system. The second cited passage (column 2, lines 47-52), as described above regarding claim 2, only describes how a computer system, suitable for implementing Maresco's invention, may operate under the control of an operating system that may execute one or more threads. Neither of the cited passages has anything to do with a secondary scheduler and certainly do not teach or suggest anything about a secondary scheduler being multi-threaded. General statements regarding multi-threaded programming and multi-threaded operating systems do not imply or suggest a multi-threaded secondary scheduler, despite the Examiner's contention to the contrary. The rejection of claim 3 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 3 also apply to claim 26.

Regarding claim 4, Cheesman in view of Maresco fails to teach or suggest wherein the secondary scheduler is executable to receive the plurality of requests from the multi-threaded application through a lock in a thread-safe manner. As shown above regarding claim 1, Cheesman in view of Maresco fails to teach a secondary scheduler receiving a plurality of requests from a multi-threaded application. The Examiner admits that Cheesman fails to disclose anything about thread-safety and relies upon Maresco,

citing column 2, lines 30-38 and column 6, lines 15-19. However, the first cited passage describes how Maresco's system "uses object-oriented programming to provide a manageable, efficient, thread-safe method for working with objects in a multi-threaded environment" (Maresco, column 2, lines 30-32). The Examiner's second cited passage only mentions creating a thread connecting a worker object in a work crew to a task and that the thread executes the task. Nowhere does Maresco mention a secondary scheduler, or anything else, receiving a plurality of requests form a multi-threaded application through a lock in a thread-safe manner. Thus, the Examiner has not provided any argument or cited any prior art that teaches or suggests wherein the secondary scheduler is executable to receive a plurality of requests from a multi-threaded application through a lock in a thread-safe manner. Furthermore, the Examiner has failed to provide any motivation for applying the worker objects, work crews and task lists of Maresco to the schedulers of Cheesman. For a more detailed argument regarding the lack motivation to combine the teachings of Maresco and Cheesman, please see the remarks above regarding the rejection of claim 1. For at least the reasons above, the rejection of claim 4 is not supported by the prior art and its removal is respectfully requested.

Regarding claim 5, contrary to the Examiner's assertion, Cheesman in view of Maresco fails to teach or suggest wherein the primary scheduler is executable to receive the plurality of requests from the secondary scheduler through a lock in a thread-safe manner. As with the rejection of claim 4, discussed above, the Examiner admits that Cheesman fails to disclose anything regarding thread-safety and relies upon Maresco. Specifically, the Examiner cites the same two passages (column 2, lines 30-38 and column 6, lines 15-19) of Maresco as cited in the rejection of claim 4. However, neither of the cited passages mentions anything regarding a primary scheduler receiving a plurality of requests from a secondary scheduler through a lock in a thread-safe manner. Instead they describe how Maresco's provides a manageable, efficient, thread-safe method for working with objects in a multi-threaded environment and how his system creates threads connecting worker objects to tasks that the threads execute. Thus, Maresco fails to disclose a primary scheduler executable to receive a plurality of requests from a secondary scheduler through a lock in a thread-safe manner. For at least the

reasons given above, the rejection of claim 5 is not supported by the prior art and removal thereof is respectfully requested.

In regard to claim 6, the Examiner's proposed combination of Cheesman and Maresco fails to teach or suggest wherein the network data resources comprise a management information server, wherein the requests comprise management requests, and wherein the multi-threaded application comprises a multi-threaded management application. The Examiner cites Cheesman at column 11, lines 5-30 and element 144a of FIG. 8, contending that Cheesman's system includes network data resources that comprise a management information server. However, the Examiner's characterization of Cheesman in view of Maresco is incorrect. Firstly, column 11, lines 5-30 of Cheesman only describe how scheduler module 144a includes a priority scheduler and a WFQ scheduler and how both schedulers allocate bandwidth for protocol data units of various queues. Nowhere does the cited passage, nor anywhere in Cheesman, mention anything about a management information server. Furthermore, the Examiner has already (in the rejection of claim 1) argued (incorrectly) that element 144a corresponds to the primary scheduler of Applicants' claims. The Examiner cannot interpret the same element (144a) as both a primary scheduler of requests and as a management information server for which the requests are scheduled.

The Examiner admits that Cheesman fails to disclose a multi-threaded manager application from which a secondary scheduler receives a plurality of management requests. The Examiner cites Maresco at column 5, lines 40-47. However, the cited portion of Maresco has nothing to do with a multi-threaded manager application from which a secondary scheduler receives a plurality of management requests. Instead, the cited passage describes a manager 200 creating a work thread to execute a task. Maresco's manager 200 is not a multi-threaded manager application from which a secondary scheduler receives management requests for network data resources. Rather, Maresco's manager 200 is the part of Maresco's system that creates and queues threads for the execution of tasks. Nowhere does Maresco describe manager 200 as sending requests for networked data resources to a secondary scheduler. One skilled in the art

would not interpret Maresco's manager 200 as a multi-threaded manager application from which a secondary scheduler receives a plurality of management requests for networked data resources.

Therefore, for at least the reasons given above, the rejection of claim 6 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 7, Cheesman in view of Maresco fails to teach or suggest a management information server coupled to the primary scheduler through a management interface, contrary to the Examiner's assertion. The Examiner cites column 11, lines 5-30 and element 144a of FIG. 8 of Cheesman. The cited passage refers only to the workings of Cheesman's Priority and WFQ schedulers of scheduler module 144a, but has nothing to do with, nor does it mention, a management information server coupled to a primary scheduler. Cheesman does not mention a management information server and does not describe any sort of management interface. Cheesman's Priority Scheduler is not coupled to a management information server, but instead is coupled to a data buffer of Cheesman's egress processor (*See, e.g.*, FIGs. 6 and 8).

Furthermore, nowhere does Cheesman or Maresco, either separately or in combination, teach or suggest wherein the primary scheduler is operable to send the requests to one or more managed objects through the management information server. The Examiner's cited passage makes no reference to sending requests through a management information server. Therefore, for at least the reasons given above, the rejection of claim 7 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 16, contrary to the Examiner's contention, Cheesman in view of Maresco fails to teach or suggest wherein executing the management requests on the managed objects comprises sending the management requests to a management information server coupled to the managed objects. The Examiner cites two portions of Maresco (column 2, lines 53-67 and column 3, lines 33-36) the first of which describes

the various type of storage media that may embody the program instructions of Maresco's manager 200, which implements Maresco's invention. This passage of Maresco clearly has nothing to do with a management information server coupled to managed objects on a network to which management requests are sent as part of executing the management requests on the managed objects. The second cited portion describes various of Maresco's objects, such as CWorkTask, CReaderTask, CWorkCrew, and CTaskQueue objects, that work together to execute a task, via a specially created thread, in Maresco's system. The second cited passage is also irrelevant to a management information server. Nowhere does Maresco's system include anything that has anything to do sending management requests to a management information server coupled to managed objects.

The Examiner is presumably contending that Maresco's manager 200 is a management information server (coupled to managed objects) to which management requests are sent as part of executing the management requests on the managed objects. However, Maresco's manager 200 has nothing to do with a management information server. Maresco's manager 200 is not coupled to managed objects. Maresco is not concerned with, nor does Maresco mention, managed objects as they are understood in the art. Maresco does not describe manager 200 as receiving management requests for managed objects.

Thus, for at least the reasons given above, the rejection of claim 16 is not supported by the prior art and its removal is respectfully requested. Similar remarks as those above regarding claim 16 also apply to claim 28.

Regarding claim 17, in contrast to the Examiner's contention, Cheesman in view of Maresco fails to teach or suggest wherein each of the management requests comprises a corresponding callback function. The Examiner cites column 8, lines 62-67 of Cheesman. However, the cited portion of Cheesman only describes how Cheesman's egress processor 114 parses protocol data units to determine the type of scheduling and queuing treatment. The cited passage has no relevance to management requests

comprising callback functions. Cheesman's egress processor does not call any callback functions, nor does Cheesman suggest anything regarding management requests comprising corresponding callback functions. Similarly, Maresco is also silent regarding the use of callback functions comprised in management requests. Thus the rejection of claim 17 is not supported by the prior art and removal thereof is respectfully requested. Additionally, remarks similar to those above regarding claim 17 also apply to claim 29.

Regarding claim 18, Cheesman in view of Maresco fails to teach or suggest receiving a response to one of the management requests from one of the managed objects after executing that management request on one of the managed objects. The Examiner cites column 9, lines 11-20 of Cheesman. However this passage does not mention anything about receiving a response to a management request from a managed object after executing that management request on the managed object. Instead, the cited passage refers only to how Cheesman's protocol data units are routed through Cheesman's queues and schedulers, but is completely silent regarding receiving a response to a management request. Maresco is also silent regarding receiving responses to management requests from managed objects.

Cheesman in view Maresco further fails to teach executing the corresponding callback function for that management request to send the response to the multi-threaded manager application. The Examiner cites column 8, lines 62-67. However, the cited passage has absolutely no relevance to executing a callback function for a management request to send the response to the multi-threaded manager application. Instead, the cited passage only refers to how Cheesman's egress processor 114 parses protocol data units to determine the type of scheduling and queuing treatment. Cheesman's egress processor 114 does not executing callback functions to send responses to a multi-threaded manager application. Nowhere does Cheesman mention anything about executing a callback function for a management request to send the response to a multi-threaded manager application.

The Examiner fails to cite any portion of either Cheesman or Maresco, either separately or in combination, that discusses a multi-threaded manager application. The Examiner cites two portions of Maresco at column 2, lines 53-67 and column 3, lines 33-36. The first passage describes the various type of storage media that may embody the program instructions of Maresco's manager 200, which implements Maresco's invention. This passage of Maresco clearly has nothing to do with a multi-threaded manager application. The second cited portion describes various of Maresco's objects, such as CWorkTask, CReaderTask, CWorkCrew, and CTaskQueue objects, that work together to execute a task, via a specially created thread, in Maresco's system. The second cited passage is also irrelevant to a multi-threaded manager application. Just because Maresco's system works in a multi-threaded environment does not teach or suggest a multi-threaded manager application.

Thus, Cheesman and Maresco, alone or in combination, do not teach or suggest receiving a response to one of the management requests from one of the managed objects after executing that management requests on one of the managed objects; and executing the corresponding callback function for that management request to send the response to the multi-threaded manager application. The rejection of claim 18 is clearly not supported by the cited art and removal thereof is respectfully requested. Arguments similar to those above regarding the rejection of claim 18 also apply to the rejection of claim 30.

Regarding claim 19, Cheesman in view of Maresco fails to teach or suggest enqueuing the response in the primary queue after receiving the response from one of the managed objects. The Examiner cites column 6, line 61 through column 7, line 10 of Cheesman. As shown above regarding the rejection of claim 18, Cheesman in view of Maresco fails to teach receiving a response to a management request from a managed object. The Examiner's cited passage of Cheesman only describes enqueuing protocol data units in the queues of Cheesman's switch fabric, but is silent regarding enqueuing a response after receiving the response from a managed object.

Cheesman in view of Maresco further fails to teach finding the callback function corresponding to the response after enqueuing the response. In fact, the Examiner fails to consider this limitation of claim 19 in this rejection at all. Neither Cheesman nor Maresco, either separately or in combination, teaches or suggests finding the callback function corresponding to the response after enqueuing the response. Additionally, as noted above, neither Cheesman's nor Maresco's systems include the use of callback functions.

Cheesman in view of Maresco also does not teach or suggest dequeueing the response from the primary queue before executing the corresponding callback function to send the response to the multi-threaded manager application. The Examiner cites column 6, line 61 through column 7, line 10 and column 8, lines 62-67 of Cheesman. As noted above, the first cited passage refers only to the queueing of protocol data units in Cheesman's switch fabric. The second cited passage only describes how Cheesman's egress processor 114 parses protocol data units to determine the type of scheduling and queuing treatment. Thus, neither cited passage, nor any other portion of Cheesman or Maresco, teaches or suggests anything regarding dequeueing a response from a primary queue before executing a corresponding callback function.

Thus, for at least the reasons give above, the rejection of claim 19 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 19 apply to claim 31.

The other dependent claims are likewise easily distinguished over the cited art. The sections of Cheesman and Maresco cited by the Examiner in regard to each dependent claim appear to have no relevance at all to the features recited in the respective claims. However, since the independent claims have been shown to be patentably distinguishable, a more thorough discussion of each dependent claim is not required at this time. However, Applicants reserve the right to make further arguments at a later date if necessary.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-48600/RCK.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,

Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: February 15, 2005